学士論文

擬似3次元表面符号による量子計算の効率化

Neary Optimal Quantum Computing
on Pseudo Three-dimentional Surface Code

慶應義塾大学
理工学部物理情報工学科

62115799　平井優我

指導教員　　山本直樹　教授

慶應義塾大学
2025 年 2 月

Contents

1 Introduction

Recent years have witnessed experimental demonstrations of fault-tolerant quantum computation (FTQC) on real devices [1][2]. These experiments highlight 2D surface codes as the most promising quantum error correction codes for FTQC due to their high physical error rate threshold. In the 2D surface code, one can perform lattice surgery [3], which enables logical operations between two distinct surface codes, each encoding a logical qubit. Additionally, to perform universal computation, one must implement magic state distillation [4] for non-Clifford gates such as the T gate or the CCZ gate. Magic state distillation is a highly costly operation; therefore, magic state cultivation [5] has recently emerged as an alternative for implementing the T gate with a cost comparable to that of a CNOT gate.

Representative quantum computing platforms include neutral atoms, superconducting circuits, semiconductors or quantum dots, and photonic quantum computers. In this paper, we study a pipeline architecture [6] for semiconductor quantum computers that utilize one-dimensional shuttling operations. A shuttling operation involves moving a qubit, such as an electron in semiconductors, to enable gate operations between two qubits that are initially far apart before the shuttling process. Neutral atom quantum computers also adopt shuttling operations, enabling the realization of a transversal CNOT gate between two distinct codes, each encoding a logical qubit.

In the pipeline architecture, we can realize a pseudo three-dimensional surface code, where 2D surface codes are stacked on top of each other. We discovered that the pseudo 3D surface code can improve routing operations via lattice surgery for gate operations, as it leverages the 3D structure for routing. In the 3D structure, it is possible to create a route in the third dimension even when no route exists on the 2D plane.

We also study the optimal placement of logical qubits made of surface codes on a processor. By using a mechanical model, such as potential energy, we nearly optimize the placement for both 2D and 3D surface code processors.

The paper is structured as follows. In Section 2, we present the notations used in the subsequent sections. Section 3 describes the stabilizer formalism, which forms the backbone of quantum error correction theory, and Section 4 explains the surface code in terms of the stabilizer formalism. In Section 5, we discuss lattice surgery operations on the 2D surface code and verify these operations using the stabilizer tableau. Section 6 introduces the pipeline architecture, while Section 7 explains the implementation of surface codes within this architecture and demonstrates how the pipeline architecture enables the realization of a pseudo 3D surface code. In Section 8, we introduce a mechanical model to optimize the placement of logical qubits on the 2D or pseudo 3D surface code. Section 9 presents the results, including improvements in routing for the pseudo 3D surface code compared to the 2D surface code, as well as placement optimization. Finally, Sections 10 and 11 provide the conclusions and future directions for the pipeline architecture and placement optimization on the pseudo 3D surface code.

2 Basics of Quantum Computation

2.1 Qubits [7]

The fundamental unit of quantum information is the qubit. Unlike a classical bit, a qubit can exist in a coherent superposition of its two states, denoted as $|0\rangle$ and $|1\rangle$. These fundamental states are physically realized as charge states in superconducting circuits, the spin of an electron in quantum dots, or atomic spin states. An arbitrary state$|\psi\rangle$ for qubits are expressed as:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \tag{1}$$

where $|0\rangle$ and $|1\rangle$ are two orthogonal basis states in hilbert space $\mathcal{H}$ and $|\alpha|^2 + |\beta|^2 = 1$. In the state vector representation, $|0\rangle$ and $|1\rangle$ are commonly expressed as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2}$$

Unlike classical processing, quantum gates acting on the Hilbert space of qubits must conserve the probability of the qubit states and are therefore unitary. we can define individual qubit gate I, X, Y and Z called pauli gate. These gates have the following properties:

$$\begin{aligned}
&I |0\rangle = |0\rangle, \quad I |1\rangle = |1\rangle, \\
&X |0\rangle = |1\rangle, \quad X |1\rangle = |0\rangle, \\
&Y |0\rangle = -i |1\rangle, \quad Y |1\rangle = i |0\rangle, \\
&Z |0\rangle = |0\rangle, \quad Z |1\rangle = - |1\rangle
\end{aligned} \tag{3}$$

where i is the imaginary unit. Thus, I is called the identity matrix and acts on a qubit trivially. From Eq. (3), one can derive the matrix representations of the I, X, Y and Z gates as:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{4}$$

If we have multiple qubits, for example, two qubits with states $|0\rangle$ and $|1\rangle$, we represent their combined state using the Kronecker product symbol $\otimes$ as $|0\rangle \otimes |1\rangle = |01\rangle$. In the same way, if we have n states of $|0\rangle$, we represent these states as $|00 \cdots 0\rangle$. For short, we denote it as $|0\rangle^n$. Additionally, we can define mutiple qubits gate G as:

$$G = \bigotimes_{i=1}^{n} P_i = P_1 \otimes P_2 \otimes \cdots \otimes P_n = P_1 P_2 \cdots P_n \tag{5}$$

where P_j is single qubit gate for jth qubit.

2.2 Gates

In Section 2.1, we introduced only the Pauli gates I, X, Y, and Z. The n-qubit Pauli gates are denoted as a group $\mathcal{P}_n$. There exist additional gates, which can be classified as either Clifford or non-Clifford gates. The definitions of Clifford gates and non-Clifford gates are given as follows.

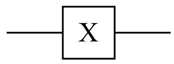
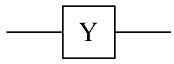
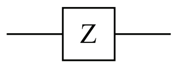
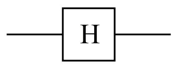
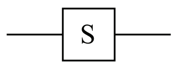
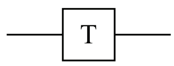
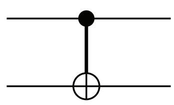
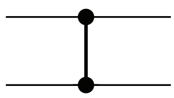
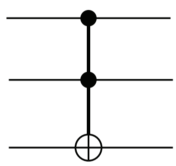
Definition 1. *A group of n-qubit Clifford gates $\mathcal{C}_n$ is defined as:*

$$\mathcal{C}_n = \{V \in U(n) \mid V \mathcal{P}_n V^\dagger \in \mathcal{P}_n\}, \tag{6}$$

where $U(n)$ denotes the n-qubit unitary group. Non-Clifford gates belong to a group disjoint from the Clifford group.

In the context of quantum computing, we often use various Clifford and non-Clifford gates to construct circuits. The circuit diagrams and matrix representations of these gates are shown in Fig. 1. An operator is synonymous with a gate in the context of quantum computing. One can verify that X, Y, Z, H, S, $CNOT$, and CZ are Clifford gates, while T and CCX are non-Clifford gates, as defined in Definition1.

Table 1

Operator	Gate (in a circuit)	Matrix
Pauli-X (X **)**		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-Y (Y **)**		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli-Z (Z **)**		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Hadamard (H **)**		$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Phase (S **)**		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
π / 8 (T **)**		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
Controlled Not ($CNOT$, CX **)**		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
Controlled Z (CZ **)**		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
Toffoli (CCX **)**		$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

When we use only Clifford gates, we perform only Clifford operations. However, it has been proven that Clifford operations can be efficiently simulated on a classical computer, as stated in the Gottesman-Knill theorem [8]. To achieve quantum supremacy, we must use non-Clifford operations for universal computation. However, such non-Clifford operations are very costly because they cannot be error-corrected in quantum error correction theory. To implement non-Clifford operations, we must perform magic state distillation [4].

3 Stabilizer Formalism [9]

In this section, we describe the stabilizer formalism, which forms the backbone of quantum error correction theory. Before delving into the main discussion, we provide a small case example of error correction and briefly introduce classical error correction theory.

3.1 Introduction to Error Correction

Noise is a great bane of information systems. Whenever we build systems for computation or other purposes, we cannot avoid noise from outside the system. However, even in such environments, classical components perform reliable computation with a failure rate typically below one error in 10^{17} operations. The details of the techniques used to protect against noise in practice are sometimes rather complicated, but the basic principles are easily understood. For example, consider the case when Alice sends Bob a single bit of information through a noisy channel, where the error rate is p. If Alice sends a bit without protection, a bit-flip error, where 0 flips to 1 and vice versa, occurs with a probability of p. As a result, Bob will fail to receive the correct information from Alice with a probability of p. In such cases, we cannot communicate reliably through the noisy channel. Yet, there is a way to address this issue. If Alice introduces redundancy for a bit of information, a process called "encoding," such as:

$$0 \to 000 \quad \text{or} \quad 1 \to 111, \tag{7}$$

Then Alice uses 2 additional bits to send a single bit of information, effectively using two additional channels. In this case, assume Alice sends 000. Each bit can be flipped with a probability of p. Bob then receives:

- No bit flipped, e.g., 000, with a probability of $(1-p)^3$,
- One bit flipped, e.g., 100, 010, 001, with a probability of $3p(1-p)^2$,
- Two bits flipped, e.g., 110, 101, 011, with a probability of $3p^2(1-p)$,
- Three bits flipped, e.g., 111, with a probability of p^3.

Thus, Bob will receive 000, 100, 010, or 001 with a probability of $(1-p)^3 + 3p(1-p)^2$. He then performs a majority vote between 0 and 1, allowing him to restore the information sent by Alice. For reliable communication through encoding, the probability of failure, $3p^2(1-p)+p^3$, must be less than 1/2. This gives the threshold for reliable communication as $p < 1/2$. Encoding a bit of information into several bits by repeating the original is called a repetition code.

3.2 Error Correction Beyond Classical Ones

In quantum environments, we cannot perform error correction in the same way as described in Section 3.1, because performing a majority vote on an encoded qubit destroys the qubit state, collapsing it into one of the eigenstates of the observable. This is crucial, as in quantum computation, we cannot ignore noise while performing operations. We need to correct errors and perform computational operations simultaneously. If we destroy the states of qubits, we cannot continue the remaining computation. There are also important differences between classical information and quantum information, requiring new ideas to make quantum error-correcting codes possible. In particular, at first glance, we encounter three rather formidable difficulties to address [9]:

- *No cloning*: One might try to implement the repetition code quantum mechanically by duplicating the quantum state three or more times. This is forbidden by the no-cloning theorem.
- *Errors are continuous*: A continuum of different errors may occur on a single qubit. Determining which error occurred in order to correct it would appear to require infinite precision, and therefore infinite resources.
- *Measurement destroys quantum information*: In classical error-correction we observe the output from the channel, and decide what decoding procedure to adopt. Observation in quantum mechanics generally destroys the quantum state under observation, and makes recovery impossible.

Fortunately, these difficulties are not fatal, and we will demonstrate a quantum version of the repetition code to illustrate this. In quantum error correction, there exist two types of errors, in contrast to classical computation: bit-flip errors and phase-flip errors. Bit-flip errors are the same as those in classical computation; for example, they are represented as $X\left|\psi\right\rangle$. Phase-flip errors occur when some qubits are acted on by Z, such as $Z\left|\psi\right\rangle$. from dualiality of X and Z, we only consider the case that bit-flip errors, and the case of phase-flip errors we can apply the same way as bit-flip. In this demonstration, we will demonstrate the repetition code for bit-flip errors. Suppose we encode the state $\left|\phi\right\rangle = a\left|0\right\rangle + b\left|1\right\rangle$ into a three-qubit space as $\left|\psi\right\rangle = a\left|000\right\rangle + b\left|111\right\rangle$. A useful way to express this is:

$$\left|0\right\rangle_L \to \left|000\right\rangle, \quad \left|1\right\rangle_L \to \left|111\right\rangle. \tag{8}$$

Here, $\left|\cdot\right\rangle_L$ is called the "logical qubit," while $\left|\cdot\right\rangle$ is called the "physical qubit." In this encoding scheme, we now obtain the encoding circuit shown below in Fig. 1.

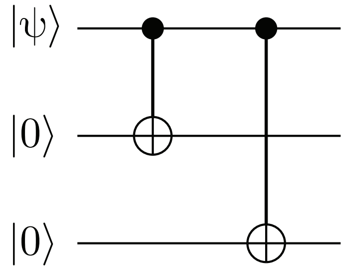


Fig. 1

Using the encoded state, we can obtain syndromes that indicate where an error has occurred. In this code, these syndromes can be determined by performing Pauli measurements of Z_1Z_2 and Z_2Z_3. For example, when no error has occurred on $|\psi\rangle$, we will obtain syndromes of 1 and 1 from Z_1Z_2 and Z_2Z_3, respectively, due to a straightforward calculation:

$$\begin{aligned} Z_1Z_2|\psi\rangle &= aZ_1Z_2|000\rangle + bZ_1Z_2|111\rangle \\ &= a|000\rangle + b|111\rangle \\ &= |\psi\rangle \end{aligned}$$

$$\begin{aligned} Z_2Z_3|\psi\rangle &= aZ_2Z_3|000\rangle + bZ_2Z_3|111\rangle \\ &= a|000\rangle + b|111\rangle \\ &= |\psi\rangle \end{aligned}$$

However, if an error has occurred on the first qubit of the three, we will obtain syndromes of -1 and 1 from Z_1Z_2 and Z_2Z_3, respectively, due to a straightforward calculation:

$$\begin{aligned} Z_1Z_2|\psi\rangle &= aZ_1Z_2|100\rangle + bZ_1Z_2|011\rangle \\ &= -a|100\rangle - b|011\rangle \\ &= -|\psi\rangle \end{aligned}$$

$$\begin{aligned} Z_2Z_3|\psi\rangle &= aZ_2Z_3|100\rangle + bZ_2Z_3|011\rangle \\ &= a|100\rangle + b|011\rangle \\ &= |\psi\rangle . \end{aligned}$$

Thus, we can determine that a bit-flip error has occurred on the first qubit and correct the error. In this case, we can correct a single error on any qubit. However, we cannot correct two or three errors. For instance, in the case of two errors on the second and third qubits of the three, we will obtain the following syndromes:

$$\begin{aligned} Z_1Z_2|\psi\rangle &= aZ_1Z_2|011\rangle + bZ_1Z_2|100\rangle \\ &= -a|011\rangle - b|100\rangle \\ &= -|\psi\rangle \end{aligned}$$

$$\begin{aligned} Z_2Z_3|\psi\rangle &= aZ_2Z_3|011\rangle + bZ_2Z_3|100\rangle \\ &= a|110\rangle + b|100\rangle \\ &= |\psi\rangle . \end{aligned}$$

These syndromes are the same as in the former case, so we cannot determine whether a single error has occurred on the first qubit or two errors have occurred on the second and third qubits. in the case of three errors, we will obtain the following syndromes:

$$\begin{aligned} Z_1Z_2|\psi\rangle &= aZ_1Z_2|111\rangle + bZ_1Z_2|000\rangle \\ &= a|111\rangle + b|000\rangle \\ &= |\psi\rangle \end{aligned}$$

$$\begin{aligned} Z_2Z_3 |\psi\rangle &= aZ_2Z_3 |111\rangle + bZ_2Z_3 |000\rangle \\ &= a |111\rangle + b |000\rangle \\ &= |\psi\rangle , \end{aligned}$$

which makes it appear as though no errors have occurred, even though errors have actually occurred. In summary, this code can detect and correct one error, can detect two errors but cannot correct them, and cannot detect or correct three errors.

One may find that if a continuous error, such as $|0\rangle \to |0\rangle + |1\rangle$, occurs, we cannot obtain syndromes. Actually, when we obtain syndromes, we perform syndrome measurements using Z_1Z_2 and Z_2Z_3. As a result, the superposition state will be projected to either $|0\rangle$ or $|1\rangle$ with a probability proportional to the coefficients of the superposition state.

In the following section, we refer to the minimum number of errors that cannot be detected as the code distance, often expressed as d, encoded qubits as logical qubits, and the number of them as k, and qubits used to encode logical qubits as physical qubits, with their number often expressed as n. The properties of the code are often denoted as $[[n, k, d]]$. In the case of the former repetition code, its properties are $[[3, 1, 1]]$, but this code cannot correct phase-flip errors. Therefore, we show how to correct both types of errors: bit-flip and phase-flip.

3.3 Stabilizer

Stabilizer formalism is the most fundamental concept in quantum error correction. The stabilizer group is denoted as $\mathcal{S}$, and its elements are Pauli matrices, so $\mathcal{S} \subseteq \mathcal{P}$, but $-I \notin \mathcal{S}$. We also define the centralizer $\mathcal{C}(\mathcal{S})$, whose elements are Pauli matrices that commute with all elements of $\mathcal{S}$. Unlike classical codes, quantum codes are designed for computation while correcting errors, so there exist logical operators that change the states of logical qubits but leave stabilizer states unchanged. These logical operators L exist in $\mathcal{C}(\mathcal{S})\backslash\mathcal{S}$. In the context of Section 3.2, $\mathcal{S} = \{Z_1Z_2, Z_2Z_3\}$ and $\mathcal{C}(\mathcal{S}) = \{Z_1, Z_2, Z_3, X_1X_2X_3\}$, so in this case, the logical operators are $L_X = X_1X_2X_3$ and $L_Z = Z_1$, Z_2, or Z_3. As a side note, Z_1, Z_2, and Z_3 are equivalent logical operators in the quotient group of the stabilizer group.

In quantum error correction theory, there are representative codes, and we will provide an example of one, called the Steane Code [10]. The Steane Code consists of 7 qubits and 6 stabilizer generators, making it an $[[7, 1, 1]]$ code. Stabilizer generators form the basis of the stabilizer group, meaning all elements of the stabilizer group can be expressed as products of stabilizer generators. stabilizer generators and logical operators are shown in Tab. 2.

One may wonder how to encode a logical qubit in such codes. Encoding involves preparing a stabilizer state for the encoded qubits, so all you need to do is perform projective measurements of the stabilizer generators. First, you initialize all physical qubits to $|0\rangle$. This state is already a stabilizer state of the Z stabilizers listed in Tab. 2. Second, you only need to perform projective measurements of the X stabilizers. Consequently, the encoded state $|0\rangle_L$ can be expressed as follows:

$$|0\rangle_L = \prod_i (I + s_i) |0000000\rangle$$

where s_i is a Z stabilizer generator, and the state is post-selected to the $+1$ eigenstates, ignoring the normalization factor. In the same way, $|1\rangle_L$ can be encoded. Thus, if you want to encode an arbitrary

Table 2

Stabilizer Generators	Logical Operators
$I\ I\ I\ X\ X\ X\ X$	
$I\ X\ X\ I\ I\ X\ X$	
$X\ I\ X\ I\ X\ I\ X$	$L_X = X\ X\ X\ X\ X\ X\ X$
$I\ I\ I\ Z\ Z\ Z\ Z$	$L_Z = Z\ Z\ Z\ Z\ Z\ Z\ Z$
$I\ Z\ Z\ I\ I\ Z\ Z$	
$Z\ I\ Z\ I\ Z\ I\ Z$	

state $|\psi\rangle = a|0\rangle + b|1\rangle$, first prepare the Bell state $a|0000000\rangle + b|1111111\rangle$ and perform projective measurements of the Z stabilizer generators. The final states is:

$$|\psi_L\rangle = \prod_i (I + s_i)|0000000\rangle + \prod_i (I + s_i)|1111111\rangle .$$

From straightforward calculations, you can confirm that $L_X|0\rangle_L = |1\rangle_L$, $L_Z|1\rangle_L = -|1\rangle_L$, and $L_X L_Z = -L_Z L_X$. By definition, a logical operator cannot be expressed as the product of stabilizer generators.

3.4 Error Detction

In this subsection, we will describe how to detect errors using syndrome measurements. Let $|\psi\rangle^t$ be the state of the physical qubits at round t, where "round" indicates the sequence number of the error correction operation. After the round t error correction, some logical operations are performed, followed by round $t+1$ error correction. Finally, this results in the round $t+1$ state $|\psi\rangle^{t+1}$. Assume there are no errors in $|\psi\rangle^t$, and that errors occur due to the logical operations between round t and round $t+1$. In this case, we obtain a syndrome m_j^t of $s_j \in \mathcal{S}$ at round t as:

$$s_j|\psi\rangle^t = m_j^t|\psi\rangle^t .$$

And at round $t+1$, we obtain a syndrome m_j^{t+1} as:

$$s_j|\psi\rangle^{t+1} = m_j^{t+1}|\psi\rangle^{t+1} .$$

In the case where the errors can be detected by s_j, m_j^{t+1} has the opposite sign of m_j^t, so $m_j^{t+1}m_j^t = -1$. On the other hand, if the errors cannot be detected by s_j, m_j^{t+1} has the same sign as m_j^t, so $m_j^{t+1}m_j^t = 1$. From this discussion, we only need to know the product of the syndromes from round t and round $t+1$.

4 Surface Code

The Surface Code, first introduced by Kitaev [11], is the most promising error correction code for quantum computing. Using this code, universal computation can be performed with magic state distillation. In this section, we will first introduce the stabilizers of the Surface Code on the torus and then on the planar surface.

4.1 Surface Code on the Torus

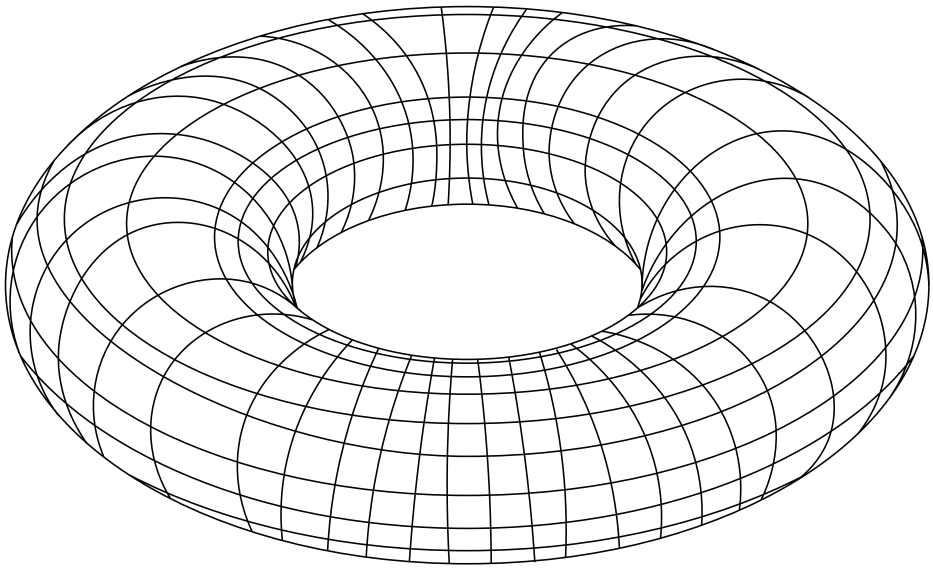


Fig. 2

Usually, the Surface Code is defined on a 1-genus torus, but it can also be defined on an n-genus torus in the same way. The lattice on the torus, shown in Fig. 2, has the following property:

$$V - E + F = 2 - 2g \tag{9}$$

where V is the number of vertices, E is the number of edges, F is the number of faces of the lattice on the torus, and g is the genus. For $g = 1$, the 1-genus case, Eq. 9 equals 0. now we introduce data qubits on the edges, X stabilizers on the vertices and Z stabilizers on the faces shown in Fig.3.

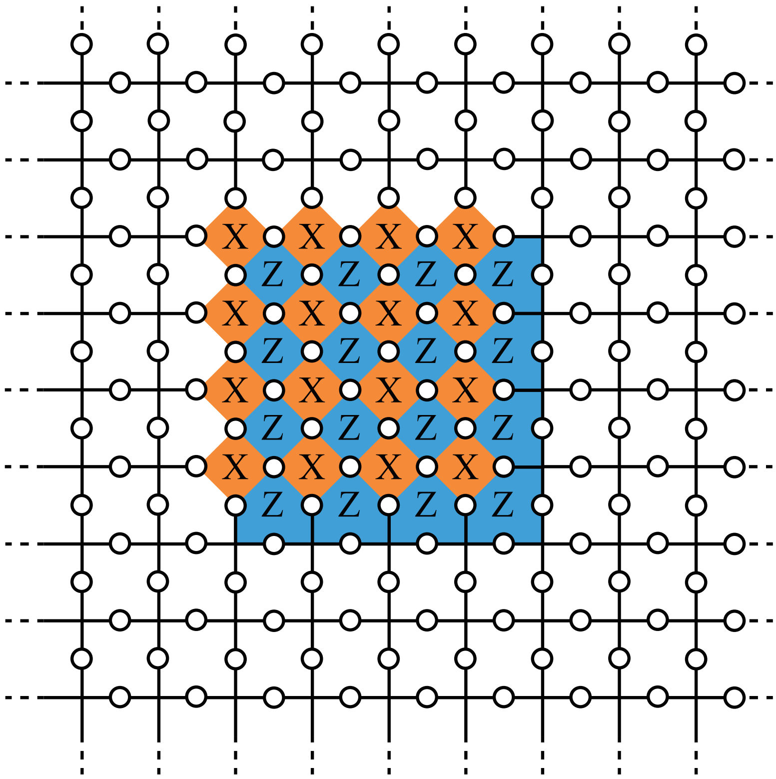


Fig. 3

In Fig. 3, we show some of the stabilizers of the Surface Code, but others exist in the remaining parts of the lattice. Thus, we have $V-1$ X stabilizer generators because the product of all X stabilizers on the vertices is the identity. From the same discussion, we have $F-1$ Z stabilizer generators. Therefore, the number of logical qubits k that can be encoded in the Surface Code is:

$$k = E - (V-1) - (F-1) = 2 \tag{10}$$

using Eq. 9. From these results, we can identify four logical operators corresponding to the non-trivial cycles on the torus. The logical operators are shown in Fig. 4, where the subscripts 1 and 2 indicate the qubit numbers of the two logical qubits. From Fig. 4, one can confirm that $L_X^i L_Z^i = -L_Z^i L_X^i$, that all logical operators commute with the stabilizers, and that the code distance is $\sqrt{n}$, which is the least weight of a logical operator.

4.2 Surface Code on the Planar

The surface code on the planar is different from that on the torus, because it has some boudaries. it is shown in Fig.5. one can find that there exists 3-weight stabilizers in the boudaries. Additionally, the surface code can encode one logical qubit.

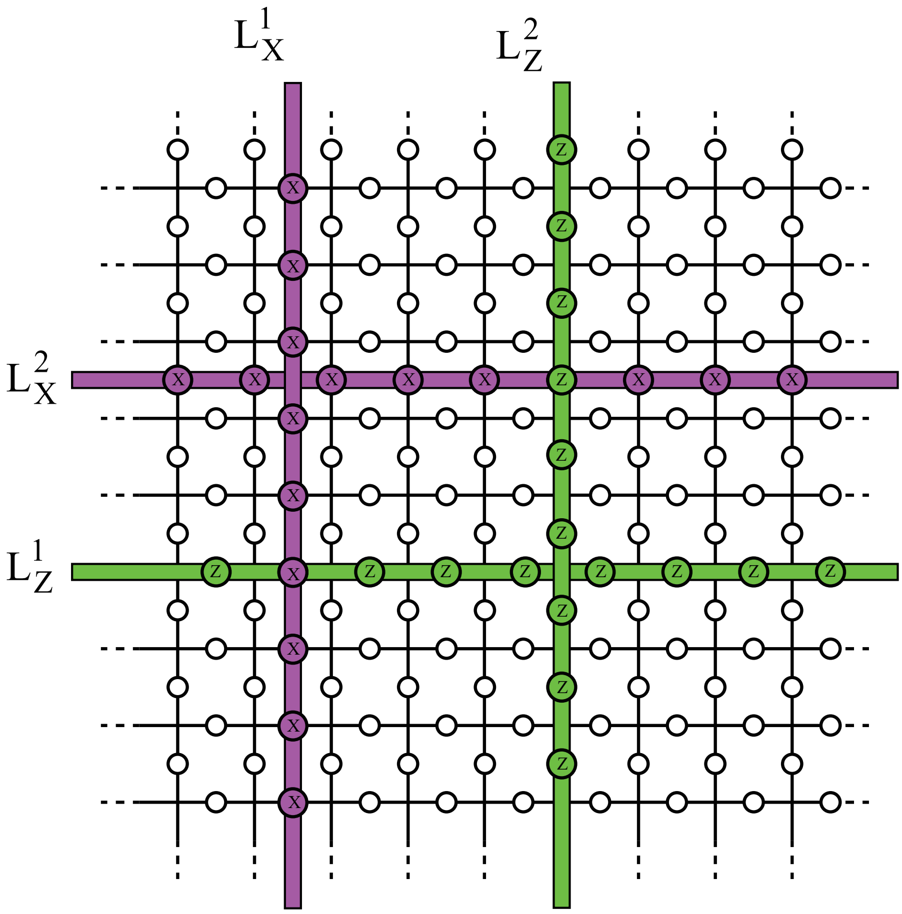


Fig. 4

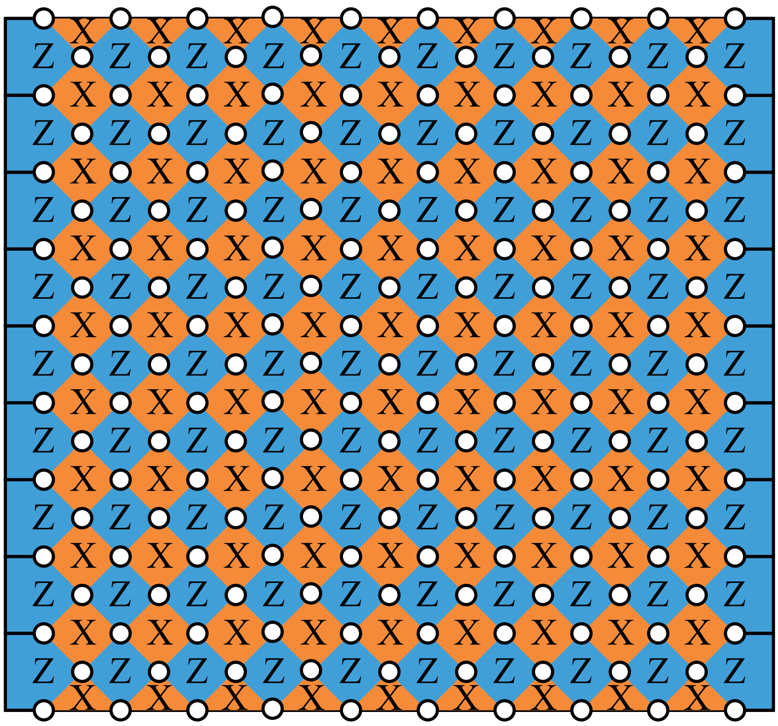


Fig. 5

REFERENCES

[1] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataides, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. "Logical Quantum Processor Based on Reconfigurable Atom Arrays", Nature **626**, 58–62. DOI: 10.1038/s41586-023-06927-3. URL: https://doi.org/10.1038/s41586-023-06927-3.

[2] Google Quantum AI and Collaborators. "Quantum Error Correction Below the Surface Code Threshold", arXiv preprint. eprint: 2408.13687. URL: https://arxiv.org/abs/2408.13687.

[3] Dominic Horsman, Austin G. Fowler, Simon Devitt, and Rodney Van Meter. "Surface code quantum computing by lattice surgery", New J. Phys. **14**, 123011. DOI: 10.1088/1367-2630/14/12/123011. URL: https://doi.org/10.1088/1367-2630/14/12/123011.

[4] Daniel Litinski. "Magic State Distillation: Not as Costly as You Think", Quantum **3**, 128. URL: https://quantum-journal.org/papers/q-2019-12-02-205/.

[5] Craig Gidney, Noah Shutty, and Cody Jones. "Magic state cultivation: growing T states as cheap as CNOT gates", arXiv preprint. eprint: 2409.17595. URL: https://arxiv.org/abs/2409.17595.

[6] Zhenyu Cai, Adam Siegel, and Simon Benjamin. "Looped Pipelines Enabling Effective 3D Qubit Lattices in a Strictly 2D Device", PRX Quantum **4**, 020345. DOI: 10.1103/PRXQuantum.4.020345. URL: https://doi.org/10.1103/PRXQuantum.4.020345.

[7] Simon J. Devitt, William J. Munro, and Kae Nemoto. "Quantum Error Correction for Beginners", arXiv preprint. eprint: 0905.2794. URL: https://arxiv.org/abs/0905.2794.

[8] Daniel Gottesman. "The Heisenberg Representation of Quantum Computers", arXiv preprint. eprint: quant-ph/9807006. URL: https://arxiv.org/abs/quant-ph/9807006.

[9] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniversary Edition. Cambridge, UK: Cambridge University Press, 2010. ISBN: 978-1-107-00217-3. URL: https://www.cambridge.org/9781107002173.

[10] A. M. Steane. "Simple Quantum Error Correcting Codes", arXiv preprint. eprint: quant-ph/9605021. URL: https://arxiv.org/abs/quant-ph/9605021.

[11] A. Yu. Kitaev. "Fault-Tolerant Quantum Computation by Anyons", arXiv preprint. eprint: quant-ph/9707021. URL: https://arxiv.org/abs/quant-ph/9707021.